International Journal on Soft Computing (IJSC), Vol.1, No.1, November 2010

A FAST GENETIC ALGORITHM FOR MINING CLASSIFICATION RULES IN LARGE DATASETS

[1]P.Vivekanandan , [2] R. Nedunchezhian,

[1]Dept. of Comp. Sci. and Engg.,Park College of Engineering and Technology,Coimbatore, India

[2]Dept of Comp. Sci. and Engg.,KalaignarKarunanidhi Institute of Technology,Coimbatore, India

[1]**Email** anandpvivek@yahoo.co.in **and** [2]rajuchezhan@gmail.com

Abstract

Nowadays data repositories are huge and are extremely large. Building a rule based classification model for these huge data sets using Genetic Algorithm becomes an extremely complex process. This is because during the learning process several passes are made over the training data set by the Genetic Algorithm and this makes it extensively I/O intensive and unsuitable. One way to solve this problem is to build the model incrementally. This paper proposes an incremental Genetic Algorithm that builds the rule based classification model in a fine granular manner by independently evolving tiny components based on the evolution of the data set which reduces the learning cost and thus making it scalable to large data sets.

Key words: *Classification, Incremental Learning, Genetic Algorithm (GA), Scalability,*

1. Introduction

Genetic Algorithm (GA) is a stochastic search method which has been widely used by the data mining community for discovering classification rules [2, 3, 4, 5, and 6]. The accuracy of the rules that GA finds are comparable and some times even more accurate than the rules obtained by the other classification algorithms[1,22]. GA shows great promise in complex domains because it operates in an iterative improvement fashion. The search performed by it is probabilistically concentrated towards regions of the given data set that have been found to produce a good classification behavior.

. In Spite of all its advantages most of the GA based classification algorithms use only a small set of training data and the task of the GA is to find out the best rule set which classifies the available instances with the lowest error rate. But today's data generated in stock market, Super markets etc are huge and have millions of records and they do not fit to the computer memory. The Mining algorithms should be made scalable to face this challenging situation successfully. In GA scalability is mainly addressed by parallel processing [14, 17] or by making the solution to converge quickly [19, 20] and there by reducing the number of generation which in turn reduces the learning time. Both are success full methods and extensive literature work is available for both the methods. They produce comparably accurate results and they also considerably reduce the Computational time.

But several incremental learning methods have been proposed in other classification methods [9, 10] to address the scalability problem. They not only reduce the costs related to the accessing of the secondary storage devices but also make the classification models to adapt to the emerging new concepts [10, 21]. Incremental learning is an untouched area with respect to Genetic

Algorithm. The proposed method in this paper tries to reduce the learning cost by incremental learning. It tries to build the model by examining very few records incrementally from the training data set and at the same time it also tries to maintain high accuracy level in par with other GA based classification methods which uses the whole training data set.

The paper is organized as follows, section 2 describes about the previous work, section 3 describes the proposed method, Section 4 describes the simulation and results and section 5 concludes the paper.

2. Previous work

Incremental learning is very popular for mining very large data sets [9, 10, 11, and 12].Very few literature employees GA as the basic algorithm for incremental learning [8]. An incremental GA was proposed by Gaun et al [8] for a dynamic environment which updates the classification rules based on the new data. Due to the arrival of new data or new attribute or class, the classification model may change. So to deal with this the author proposes an incremental based GA. GABIL (Genetic Algorithm Based Incremental Learning) proposed by De Jong et al [13] is an incremental learning algorithm which learns as the data evolves. When a new record is misclassified, it uses all the records of the training set again including the new record to form the new rule set. But the above algorithms do not address the scalability issue with respect to size.

In the literature complexity arising due to scalability issue is also addressed by parallel processing using GA as a basic algorithm (14, 17). For example Parallel Genetic Algorithm proposed by D. L. A Araujo et al [14] addresses the scalability issue with respect to GA. It involves multiple processors and the data set is divided into multiple parts (data sets). The multiple data sets are distributed to multiple processors and each processor generates rules for each data set. The rules generated by each processor are again shared by all processors for fitness calculation.

Scalability is also addressed in another angle namely quick convergence by making changes in the GA operators (18, 19, and 20). For example Zhijiang Jiang (20) et al proposes a hybrid Genetic Algorithm (HGA) for heterogeneous computing environment scheduling. In their proposed method crossover and mutation probabilities are adjusted adaptively with respect to average fitness value of the population. They argue that their proposed Genetic Algorithm does not get stuck at a local optimization easily, and also it is fast in convergence.

3. Proposed method

In the proposed method the scalability issue is addressed in a different perspective with respect to GA. In classification problems GA should use all the records in the training set to calculate the potential solutions fitness values and for very large data sets this constitutes the most part of the learning cost. By minimizing the number of records to be examined proportionate learning cost can be saved. The core idea of the paper is to make GA to learn incrementally and because of it the number of records needed to build an accurate model can be reduced considerably.To reduce the learning time with respect to data sets that do not fit to the main memory, normal procedure followed by the data mining community is to partition the data set and mine incrementally. The simple GA based partitioning algorithm is proposed by the algorithm PGA (). It is a two pass algorithm. It divides the training data base D into p partitions and GA is applied to each partition and the rules sub set Li is mined for each partition. Then the rules sub sets are combined together to form a candidate rule set C. In the second pass the support and confidence of the candidate rules are computed using the whole data set D and the rules whose support and confidence above some threshold is produced as the final rule set R.

```
Algorithm PGA()
    1. Divide D into partitions D1,D2,…,Dp;
    2. For I = 1 to p do
    3. Li = GA(Di);  // apply GA
    4. C = C ∪ Li;
    5. end for
    6. use  C on D to generate R;
```

Partitioning method reduces the learning time considerably particularly when the data set is too large and does not fit into the main memory. But some times all the partitions need not be processed because the partitions may be similar to each other and when mined will yield same set of rules. In large data sets the probability of similarity between the partitions is high. Finding them and dropping the partitions which are similar to some other partition, whose rules have been already mined will reduce the learning cost considerably.

Let there be p partitions in a training data set and all the partitions be of equal in size. Let their processing time also be equal and let it be t for each partition.

Learning time(Lt)= $n*t$ Equation (1)

$\Rightarrow Lt$ ά n Equation (2)

Learning time is proportionate to n (number of partitions processed) and n will equal to p if all the partitions are processed.

The similarity between partitions can be found based on their difference in misclassified record count when examined by a common rule set. The definition for misclassified records is defined by definition 1 and Percentage of Misclassified record (PMRC) for a partition is defined by definition 2.

Definition 1: (Misclassified Records). A Record is called as a misclassified record if it has no matching rule in the given rule set.

Definition 2: (Percentage of Misclassified Record – PMRC). It is the percentage of records in a partition that are misclassified with respect to a rule set.

Let R be a rule set and P1 be a partition with r records. Let M1 be number of misclassified records in P1 with respect to R.

$PMRC (P1) = (M1/r)*100$ Equation (3)

If two partitions P1 and P2 are said to be similar if
$|PMRC (P1)-PMRC (P2)| < \varepsilon$ and ε is the similarity threshold defined by the user.

We can also examine a partition by its parts and independently build the rule set for each part by employing GA. In that process if we can able to drop some part of the partition without examining will reduce the learning cost with respect to that partition. This is because repeated processing of the records by GA for fitness calculation contributes much to the learning cost. Reduction in learning cost of partition will have an impact on the overall learning cost. Let t1 be the time taken to process a record in a partition and let the time taken to process all the records are same. Let a partition contains r records

Processing time (t) of a partition = r*t1
Equation (4)

=> t ά r Equation (5)

Processing time of a partition is directly proportionate to number of records in a partition. To restrict the number of records to be examined the partition can be divided into sub blocks based on their class label. Similarity between sub blocks of different partitions can be easily identified and then the duplicate sub blocks can be dropped without examining and this will reduce the learning cost of a partition substantially.

Attribute Values	**Class**	**Sub block name**
a1 b2 **a1 b2**	**C1** **C1**	**Sp11**
a3 b3 **a3 b3**	**C2** **C2**	**Sp12**
a2 b4 **a2 b4**	**C3** **C3**	**Sp13**

Table 1

Consider the partition P1 described in table 1. It contains six records with two attributes a and b and a class label c. Let it be divided into three sub blocks sp11, Sp12, Sp13 based on the class label. GA is applied to P1 and the rules mined will be a1, b2→c1, a3, b3→c2, a2, b4→c3.

Attribute Values	**Class**	**Sub block name**
a1 b2 **a1 b2**	**C1** **C1**	**Sp21**
a3 b3 **a3 b3**	**C2** **C2**	**Sp22**
a2 b1 **a2b1**	**C3** **C3**	**Sp23**

Table 2

Consider the partition P2 described in table 2 which occurs next to P1. It is also divided into three sub blocks sp21, **s**p22, **s**p23 based on the class label. The rules mined by applying GA to P2 will be a1, b2→c1, a3, b3→c2, a2, b1→c3.

Comparing the rule sets of P1 and P2 it can be noticed that only the last rule differs. This is because most of the records in P1 and P2 are similar and only sub blocks sp13 and sp23 differs. Let P1 occurs before P2 and sub blocks (sp11, sp12, sp13) of P1 are examined separately by GA and there by the rules of classes' c1, c2 and c3 are generated independently. Then instead of using entire P2 block for mining in the next iteration it is enough to mine only the last sub block sp23 of P2. This is because the sub blocks sp21 and sp22 are similar to sp11 and sp12 of P1 respectively and mining the rules by using them will produce only duplicate rules which have been already mined.

So based on the above discussion it can be understood that, building the rules of each class independently as a tiny component in an incremental fine granular manner helps in achieving scalability. It is because the similarities between sub blocks of classes can be identified easily based on the misclassified record percentage as described earlier and the duplicate sub blocks can be dropped without examining.

3.1 Overview

Let D be the data set. Let D be partitioned into p partitions such that each partition fit to the main memory. Let there be n classes and rules are to be mined for predicting all the n classes. The rules are defined by rule set R and it is divided into R1 R2 ---- Rn independent rule sets for each class and are all learned in an incremental independent manner. Initially they are all empty.

Each partition of the training data set is examined sequentially and the records in the partition are divided into sub blocks based on the class labels of the records. Rules are mined for each sub block i (ie for each class) separately and are added to their respective rule set (Ri). As discussed in the above section, before mining, each sub block should be checked whether it is similar to any other sub block of a previous partition. If it is similar then it will not yield any new rules so it can be dropped without examining. The basic measure used for this comparison is Current Percentage of Misclassified Record Count (CPMRCi) defined by definition 3.

Definition 3 (Current Percentage of Misclassified Record Count- CPMRCi). It is the misclassified percentage of a class i with respect to the data partition that is being processed and is based on the rules generated so far for that particular class (Ri). It is calculated for all the classes (i ranges from 1 to no of classes) based on their corresponding sub blocks when the partition is initially copied to the memory.
CPMRCi= no of misclassified records of a sub block whose class label is i*100/Total number of records of the sub block. Equation (6)

To find out the similarity of a sub block it must be compared with a previously mined sub block of the same class and it is called as reference sub block for that particular class. Its CPMRCi value is called as Reference Percentage of Misclassified Record Count(RPMRCi) and It is defined by the definition 4.

Definition 4 (Reference Percentage of Misclassified Record Count-RPMRCi). It is the lowest CPMRCi so for obtained for the class i.

Initially for each class i
RPMRCi=100 Equation (7)

After processing each partition all the RPMRCi values are compared with their corresponding CPMRCi values and the greater values are replaced by its CPMRCi value.
RPMRCi=CPMRCi if RPMRCi>CPMRCi fo each class i

Before mining the rules of a sub block of a partition its CPMRCi value is calculated. The difference between the CPMRCi and RPMRCi values of a particular class is the Misclassified accuracy difference (MADi). It is defined by the definition 5.

Definition 5(Misclassified Accuracy Difference- MADi): The difference between CPMRCi and RPMRCi values of a particular class i with respect to the current partition that is under process.

MADi= RPMRCi-CPMRCi Equation (8)

Definition 5(Misclassified Accuracy Difference threshold MADTi): The similarity difference threshold is proposed by the user for each class i. When the Misclassified Accuracy Difference is less than this threshold then the sub block in the current partition is dropped without examining.

If the MADi is greater than certain threshold (MADTi) it indicates that some new rules have emerged and it is to be mined. If the difference is very low indicates that the data set for that particular class is repetition and even if we try to mine rules it may not generate any new rule and so it can be dropped. This greatly reduces the cost of examining the records whose rules are mined already. This in turn reduces the over all learning cost.

3.2 Proposed ISGA Algorithm

. Two phases are there in the proposed method. First is the rule learning phase in which potential candidate rules are mined and in the second phase predictive accuracy and support of all the candidate rule sets are calculated and the rules whose support and confidence less than certain threshold are removed. Duplicate and overlapping rules are pruned and rest of the rule set is produced as the output. The algorithm for the first phase find_rule() is described below:-

Algorithm Find_ Rule()

```
1.  Input: Example data setsD1. .Dj…Dm divided into m blocks
2.  output: A classification rule sets R1…Ri…Rn for all the class labels 1 to n
3.  Variables: IRi…IRn intermediate rule sets
4.  Wi…..Wn Windows for classes 1 to n
5.  for i=1 to n
6.     RPMRCi=100
7.  end loop i
8.  for j= 1 to m
9.  Read Dj
10. Examine class label of all the records r1,r2--- of Dj and add to their corresponding window Wc if they
    belong to class c.
11.   for i= 1 to n
12.   CPMRCi=Find_misclassified_ percent(Wi,Ci)
13.        if (CPMRCi-RPMRCi>MADTi)
14.          IRi=Simple Genetic Algorithm (Wi,Ci)
15.          Ri=Ri U IRi
16.         CPMRCi=Find_ misclassified_ percent(Wi,Ci)
17.       end if
18.       if (CPMRCi<RPMRCi) then RPMRCi=CPMRCi
19.       IRi=null
20.   end for i
21.  Empty all Windows Wi to Wn
22.  end  for j
23.  return Rule set R(R1,R2,--Rn)
```

In the first phase first all the RPMRC values of all the n classes are initialized to 100. Then the partitions of the training data set are read on by one and are processed sequentially. When a partition is brought to the main memory the class c of each record of that partition is determined and is added to a separate sub block Wc (a window) based on its class label c. The Find_misclassified_percent() call in the above algorithm finds out the CPMRCi value of a class i for the current data block Dj. It is compared with the RPMRCi value of the corresponding class and if the difference between them is greater than the threshold MADTi, GA is applied to the corresponding sub block Wc. The Simple_Genetic_algorithm call in the find_rule () does this job. The mined new rules are added to the rule set Ri of the class i. Again the Current Percentage of Misclassified Record Count (CPMRCi) value of the class i is updated based on the new rule set. If the CPMRCi value for a class i is lower than the RPMRCi value of that particular class, then the corresponding RPMRCi value is replaced by the new lower value. The process is repeated for all the classes 1 to n for a partition and also for all the partitions.

In the second phase the full data set is used for calculating the predictive accuracy of all the rules that has been mined and rules with particular predictive accuracy (say above 70%) and support (say above 5%) is retained and rest discarded. Unwanted and overlapping rules are pruned and thus the classification model is built.

3.3Genetic Algorithm

The Encoding used in the GA is a fixed length encoding containing one bit for each value of the attribute. By an extra flag bit for each attribute it is made to represent variable length rule [7]. The potential solutions are encoded as chromosomes and this is called population. The population is processed in loop using selection, crossover, mutation, insertion and remove operations such that the best solution gradually emerges. Crossover is performed by exchanging the genes including flag bit with a probability called crossover probability (P*c).* The Mutation operator changes the value of an attribute to another random value selected from the same domain with a probability called mutation probability (Pm*).* Insert and Remove operators [7, 15] control the size of the rule. Insert operator activates the gene by setting its flag bits and Remove operator deactivate a gene by resetting the flag bits with a varying probability (Pi and Pr) and the probabilities range from 0 to 30% based on the number of genes that take part in the rule. Best Elite_percent of chromosomes are considered as elite and they are copied to the next generation unaltered.

The fitness function comprises of two components, namely a Predictive Accuracy and Comprehensibility. The rules are of the form IF A1 $\wedge$ A2…An THEN C. The antecedent part of the rule is a conjunction of conditions say A (conjunction of A1, A2…An). A very simple way to measure the Predictive Accuracy is

Predictive Accuracy (PA) = $(|A\&C|-1/2)/|A|$ Equation (9)

where $|A|$ is the number of examples satisfying all the conditions in the antecedent A and $|A\&C|$ is the number of examples that satisfy both the antecedent A and the consequent C. Intuitively, this metric measures Predictive Accuracy in terms of how many cases both antecedent and consequent hold out of all cases where the antecedent holds. The term ½ is subtracted to penalize the rules covering few training examples [7].

The standard way of measuring Comprehensibility [16] is to count the number of condition in the rule. If a rule has at most L condition, the Comprehensibility of the rule (or individual) p can be defined as

Comprehensibility(CM)= $(L-n)/(L-1)$. Equation(9)

Where n is the length of the rule (or individual) p.

The fitness function is computed as the arithmetic weighted mean of Comprehensibility and Predictive Accuracy. The fitness function is given by:

Fitness=$W1$*PA+$W2$*CM Equation (10)

Where Wi *is* the weight defined by the user.

When GA is applied to a sub block a new fitness function is used as described in the equation (11) to identify the potential rules of the particular sub block. Potential rules final fitness values are calculated based on the predictive accuracy as defined in the equation (11) before adding it to the rule set R.

Fitness=$W1$*(|A&C|/|C|+$W2$*CM Equation (11)

Where |C| is the number of training records in the corresponding sub block.

4. Simulation

4.1. Description of the dataset

The simulation was performed using the data sets obtained from the UCI machine repository (http://www.ics.uci.edu/). These data sets are normally used as a benchmark for evaluating algorithms performing classification task.

4.1.1 Data sets:

Car evaluation data set contains 1728 records and 6 attributes. All attributes are categorical and are listed in table 1. The target class attribute has four values namely ‘unacc’, ‘acc’, ‘good’, ‘vgood’. To generate larger data sets of size 10000, 20000 and 30000 the records are duplicated and randomly arranged such that the data distribution is proportionately similar to the original data set.

Attributes	Values
Buying:	Vhigh, high, med, low.
maintenance	Vhigh, high, med, low.
Doors	2, 3, 4, 5more.
Persons	2, 4, more.
Lug boot:	Small, med, big
Safety	Low, med, high.

Table 3 Car Evaluation Data set attributes

Further to generate sufficiently large data sets six data sets described in table 4 are used. The data sets are duplicated and randomly arranged to form a data stream containing one lakh records.

4.1.2 Results

The experiments were performed on a Pentium4 (256MB RAM) windows XP System and the algorithm was developed using java. The proposed method (ISGA) is compared with the Simple Genetic Algorithm (SGA) (7) and a Parallel Genetic Algorithm which is implemented by using 4 processors. For ISGA, some of the related parameters were set as mentioned: P_c=0.8, P_m=0.05, P_i=[0,0.3], P_r=[0, 0.3],

Index	Data Set	No of Instance	No of attributes	No of classes	No of Instances	No of repetition
1	Yeast	1389	8	10	1,00,000	68
2	Nursery	12960	8	5	1,00,000	8
3	LED display	10000	7	10	1,00,000	10
4	Page Blocks	5473	10	5	1,00,000	19
5	Australian Sign language	12546	8	3	1,00,000	8
6	Balance	625	4	3	1,00,000	160

Table 4 Data set Description

Data set	Error rate (%)			Run Time (s)		
	SGA	ISGA	Parallel SGA	SGA	ISGA	Parallel SGA
Yeast	41.6	40.9	40.6	4664.2	670.1	1182
Nursery	14.30	11.3	11.3	2332.5	331.4	472.8
LED display	27.57	28.54	28.54	4081	579.2	1034.5
Page Blocks	3.2	3.2	3.2	2915.3	421.5	738.7
Australian Sign language	16.19	14.67	14.37	1399.2	197.1	354
Balance	8.4	8.2	8.2	699.6	99.2	177.3

Table 5 Results

$W1 = 0.7$, $W2 = 0.3$ and the data partition size is 500 records per block. The misclassification accuracy difference threshold (MADTi) is kept as five percent for all the classes. The experiment is performed for Car evaluation data set using training data set of different sizes (10000, 20000 and 30000).

Population size for the SGA algorithm is 100 and it is executed for 100 generations. For the Parallel GA the data set is divided into four parts and is used by four processors to mine the rules. The population size for each processor is 100 and number of generations is also 100.The Execution time and the accuracy of the rules obtained was monitored. The results of the Car Evaluation data set is described by figure 1 and 2. While for the rest of the data sets the results are tabulated in table 5.

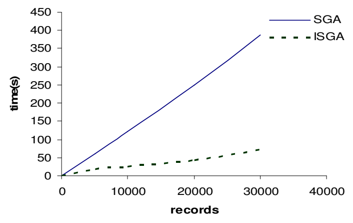

Figure 1 Comparison of time for various sizes

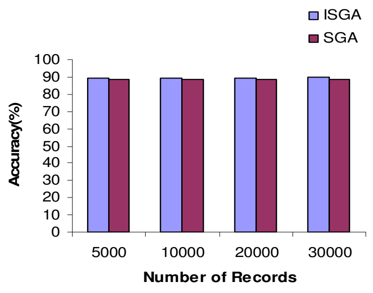

Figure 2: Comparison of accuracy

4.2.1 Execution Time

It can be noticed from figure 1 and table 5 that the proposed methods execution time is approximately 6.9 times faster when compared to the Simple Genetic Algorithm and 1.7 times faster than the Parallel Genetic Algorithm which has been implemented by using four processors. This indicates that the proposed method reduces the learning cost to a great extent. This makes the proposed method more suitable for huge data sets.

4.2.2 Accuracy

The accuracy of the rules mined by the proposed method is better than or similar to that of the rules mined by the simple genetic algorithm (Figure 2, Table 5). So it can be concluded that the proposed method produces accurate rules comparable with the Simple Genetic Algorithm and Parallel Genetic Algorithm by examining less number of training records.

5. Conclusion

To make the Genetic Algorithm applicable for generating rules for large data sets it should be made scalable and for this purpose an incremental Genetic Algorithm is discussed in this paper which builds the model in a fine granular way. It considers each rule set for a class as a tiny component. It mines and adds rules independently to these components when ever the data distribution changes. This avoids unnecessary mining activity there by reducing learning cost and this makes the algorithm scalable with respect to size. The efficiency of the algorithm is proved by using standard data sets. In the future work attempt will be made to reduce learning time further so that GA can be made suitable for fast data streams.

References

1. Linyu Yang, Dwi H. Widyantoro, Thomas Ioerger and John Yen, "An Entropy-based Adaptive Genetic Algorithm for Learning Classification Rules", *Proceedings of the 2001 Congress on Evolutionary Computation,* , Issue , 2001 Page(s):790 – 796,2001.

2. K.A. De Jong., W.M. Spears and D.F. Gordon, "Using genetic algorithms for concept learning", *Machine Learning,* volume 13, page(s) 161-188, 1993.

3. C.Z. Janikow, "A knowledge-intensive genetic algorithm for supervised learning", *Machine Learning,* volume 13, page(s) 189-228, 1993.
4. D.P. Greene and S.F. Smith, "Competition-based induction of decision models from examples", *Machine Learning*, volume 13, page(s) 229-257, 1993.
5. Poonam Garg "A Comparison between Memetic algorithm and Genetic algorithm for the cryptanalysis of Simplified Data Encryption Standard algorithm", International Journal of Network Security & Its Applications (IJNSA), Vol.1, No 1, April 2009
6. E. Noda, A.A. Freitas and H.S. Lopes, "Discovering interesting prediction rule with a genetic algorithm ", *Proceedings of the 1999 Congress on Evolutionary Computation,* Volume 2, 1999.
7. Dehuri, S., Mall, R. "Predictive and Comprehensible Rule Discovery Using A Multi-Objective Genetic Algorithm", *Knowledge Based System,* volume 19, pp: 413-421, 2006 (SCI).
8. S.U. Guan and F.ZhuCollard, "An incremental approach to genetic-algorithms based classification. Systems", *Man and Cybernetics, Part B, IEEE Transactions*, volume *35*, no. 2, page(s) 227 – 239, 2005.
9. Domingos, P., Hulten, G., "Mining high-speed data streams", *Proceedings KDD* 2000, ACM Press, New York, NY, USA, pp. 71–80. 2000
10.Hulten, G.,Spencer, L.,Domingos, " Mining time-changing data streams", *Proceedings KDD* 2001, ACM Press, New York, NY, pp. 97–106. 2001
11. Polikar, R. Upda, L. Upda, S.S. Honavar, " Learn++: an incremental learning algorithm for supervised neural networks ", *IEEE Transactions on Systems, Man, and Cybernetics,*Volume 31, Issue:4 On page(s): 497-508 . 2001
12. Jing Gao, Bolin Ding, Wei Fan, Jiawei Han,Philip S.Yu, "Classifying Data Streams with Skewed Class Distributions and Concept Drifts"**,** *IEEE Internet Computing, Special Issue on Data Stream Management*(IEEEIC),Nov/Dec. 2008, page(s)37-49, 2008.
13.Kenneth A. De Jong , William M. Spears, "Learning concept classification rules using genetic algorithms", *Proceedings of the 12th international joint conference on Artificial intelligence,* p.651-656, August 24-30, Sydney, New South Wales, Australia. 1991.
14. D. L. A Araujo., H. S. Lopes, A. A. Freitas, "A parallel genetic algorithm for rule discovery in large databases" , *Proc. IEEE Systems, Man and Cybernetics Conference,* Volume 3, Tokyo, 940-945, 1999.
15 Wojciech Kwedlo, Marek Kretowski, " Discovery of Decision Rules from Databases: An Evolutionary Approach" *Second European Symposium, PKDD* '98, Nantes, France, September 23-26, 1998.
16. Xian-Jun Shi Hong Lei , " A Genetic Algorithm-Based Approach for Classification Rule Discovery". *International Conference on Information Management, Innovation Management and Industrial Engineering, 2008***,** Volume: 1, page(s): 175-178, 2008.
17.Abhishek Verma, Xavier Llorà, David E. Goldberg, Roy H. Campbell, "Scaling Genetic Algorithms Using MapReduce,", *Ninth International Conference on Intelligent Systems Design and Applications*, pp.13-18, 2009.
18 Dirk Thierens "Scalability problems of simple genetic algorithms" Evolutionary Computation Volume 7 , Issue 4 Pages: 331-352 1999.
19. Alfonsas Misevicius., "A fast hybrid genetic algorithm for the quadratic assignment problem", *Proceedings of the 8th annual conference on Genetic and evolutionary computation,* Pages: 1257 – 1264, 2006.
20. Zhijiang Jiang , Shengzhong Feng , "A Fast Hybrid Genetic Algorithm in Heterogeneous Computing Environment", *Fifth International Conference on Natural Computation* pages :71-79 2009
21. Sattar Hashemi, Ying Yang, Zahra Mirzamomen, and Mohammadreza Kangavari, "Adapted One-versus-All Decision Trees for Data Stream Classification", *IEEE Transactions On Knowledge And Data Engineering*, Vol. 21, No. 5, May 2009
22. T. Venkat Narayana Rao et al., " / Genetic Algorithms and Programming-An Evolutionary Methodology", *International Journal of Computer Science and Information Technologies*, Vol. 1 (5), 427-437, 2010.